SYSTEM AND METHOD FOR ENERGY EFFICIENT DATA PREFETCHING

The present invention relates generally to the performance enhancement of prefetching in a network environment and particularly to energy-efficient prefetching of files in a network environment.

BACKGROUND OF THE INVENTION

In networked environments, such as the Internet or communications networks, fetch latency is inevitable. Fetch latency is the time lag between when an object (e.g., a document or file) is requested and when the object is received. In a computer network environment, fetch latency is often referred to as page fetch latency. Page fetch latency is measured by the time lag between when users of client computer systems click on a page link to when the page actually appears on their screens. Page fetch latency may vary depending upon many factors, including available bandwidth.

Two responses to high page fetch latency in the World Wide Web ("WWW") context are the use of mirroring and caching. Mirroring reduces fetch latency. However, mirroring requires manual target selection, which can be difficult to administer. Caching is typically implemented by having a proxy server store (i.e., cache) copies of frequently accessed objects. The proxy server, which may also be called a cache server, is typically located near one or more client computers, and is used to reduce network load. User requests for objects are initially directed to the proxy server, instead of the web servers at the network addresses specified in the user requests. If the object specified by the user request is in the proxy server, the proxy server sends the object to the requesting client without accessing the web server at the specified network address. Caching reduces fetch latency for repeatedly accessed objects, but does not improve first retrieval.

The world wide web allows caching at multiple points, so a cache mechanism is also typically implemented as part of the user agent program (e.g., a web browser) that runs on a client computer. Alternatively, or in addition, a proxy cache server can be implemented as an

application or process that runs on the client computer, separate from the user agent program. The use of a proxy cache server located on the same computer as the user agent program is sometimes used to extend the functionality of the user agent program without directly modifying that program, while preserving the benefits of locating the cache on the same computer.

Prefetching can also be used to avoid, or at least to reduce, fetch latency. Prefetching is a technique whereby a client predicts a future request for a file on a remote server, fetches (i.e., prefetches) the file in advance, and stores the prefetched file in a local cache. Thus, the file is stored in the local cache before the client makes an explicit request for it. Access by the client to the local cache is much faster than access to the remote server. Thus, if the client does indeed request the file as predicted, the fetch latency is less than it would have been if the client requested the file from the remote system without prefetching.

Prefetching can improve retrieval times for even first-time page fetch requests, which is not possible with caching alone. However, prefetching can also result in the waste of system resources when a prefetched file is not requested by the user during the time that the prefetched file remains in the client computer's cache. Prefetching can even result in an increase in fetch latency when the prefetching of files interferes with demand fetching, which is the fetching of files actually requested by the user. More generally, the extra load imposed on client, server and network resources by prefetching may degrade performance of one or more of the client, server and network, especially if a large number of files are prefetched and the accuracy of the prefetching is low.

The present invention is based, in part, on the observation that energy efficiency is not a factor considered in prior art prefetching schemes. Energy efficiency is particularly important in battery powered, portable devices, and is also important in other contexts. Prefetching may increase energy usage by the client, server and/or network because the total number of files fetched is increased by the prefetching, since at least some of the prefetched files will not be used by the client computer while the files remain in the client computer's cache. The lower the accuracy of the prefetch predictions, the higher the energy usage will be. However, there may also be energy usage efficiencies associated with prefetching, because fetching multiple

files in a burst may be more energy efficient than fetching the same files one at a time with periods of inactivity between the individual file fetches. More generally, there is a tradeoff between energy efficiency and the extent to which average fetch latency is decreased through the use of prefetching.

It would therefore be advantageous to provide an energy-efficient prefetching system and method that improves user-perceived network performance with prefetching.

SUMMARY OF THE INVENTION

In summary, the present invention is a system and method for performing energy efficient data prefetching in conjunction with a client computer system. The client computer system uses a prefetch prediction model having energy usage parameters to predict, or otherwise take into account, the impact of prefetching specified files on the system's energy usage. A prefetch prediction engine utilizes the prefetch prediction model to evaluate the specified files with respect to prefetch criteria, including energy efficiency prefetch criteria, and generates a prefetch decision with respect to each file of the specified files. For each specified file for which the prefetch prediction engine generates an affirmative prefetch decision, an identifying entry is stored in a queue. The client computer system fetches files identified by entries in the queue, although some or all of the entries in the queue at any one time may be deleted if it is determined that the identified files are no longer likely to be needed by the client.

In an other aspect of the invention, a server computer includes a server module for responding to a request from a client computer for a specified file and for generating a reply to the request. The reply includes a content portion comprising the specified file and a supplemental portion distinct from the content portion. A prefetch predictor identifies additional files for possible prefetching by the client computer. The server module includes in the supplemental portion of the reply to the request from the client computer prefetch hint information identifying at least one of the additional files.

BRIEF DESCRIPTION OF THE DRAWINGS

Additional objects and features of the invention will be more readily apparent from the following detailed description and appended claims when taken in conjunction with the drawings, in which:

Fig. 1 is a block diagram of a client computer system with a prefetch prediction engine and prefetch prediction model in accordance with a preferred embodiment of the present invention.

Fig. 2 depicts the data structure of a server's response to a request by a client computer in accordance with an embodiment of the present invention.

Fig. 3 is a block diagram of a prefetch prediction model.

Fig. 4 is a block diagram of system components interoperating with a prefetch prediction engine.

Fig. 5 is a flow chart of an energy efficient prefetching method performed by a client computer.

Fig. 6 is a block diagram of a server computer system with a prefetch predictor and prefetch prediction model in accordance with a preferred embodiment of the present invention.

Fig. 7 is a block diagram of a server-side prefetch efficiency model.

Fig. 8 is a block diagram of a server module having a prefetch hint pruner as well as a prefetch predictor and prefetch efficiency model.

Fig. 9 depicts the data structure of a client's request message to a server in accordance with an embodiment of the present invention.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

Referring to Fig. 1, a client computer system 100 is shown in accordance with the present invention. The system 100 includes memory 108 for storing applications executable by one or more central processing units 102 and data structures such as queues and databases. A user interface 104 allows input by and output to a user of the system. A network interface 106 allows communication through communications network(s) 140 with servers (server computer systems) 220. The network 140 may be the Internet or other wide area network, an intranet, a local area network, a wireless network, or a combination of such communication networks.

In one embodiment, memory 108, which typically includes high speed random access memory as well as non-volatile storage such as disk storage, stores:

- an operating system 110, for providing basic system services;
- file system 112, which may be part of the operating system;
- application programs 114;
- a browser 116, for viewing and requesting documents, such as documents downloaded from servers via the Internet or an intranet;
- a prefetch prediction model 118, which includes energy usage parameters (see Fig. 3) for predicting the impact of prefetching specified files on energy usage by the system;
- a model updater 120, which is a software module for updating the prefetch prediction model 118 based on energy efficiency and utilization of previously prefetched files;
- a prefetch prediction engine 122, which determines, based on the prefetch prediction model 118, which files of a set of prefetch candidates should be prefeteched;
- fetch queues 126, including a demand fetch queue 128 and a prefetch queue 130;
- a scheduler 132 for scheduling downloads of files listed in the demand fetch queue 128 and prefetch queue 130; and
- a cache 134 for storing copies of previously downloaded files, including prefetched files.

Memory 108 may also optionally store a battery meter procedure 124 for receiving information concerning the remaining energy stored in a battery. The remaining energy is

sometimes herein called the "current energy supply level." Many of the features of the present invention are not necessarily distinct applications. For example, the prefetch prediction engine 122 and model updater 120 can be implemented using a single software application that implements their joint functionality. In another example, the prefetch prediction model 118 may be implemented as part of the prediction engine 122.

The interaction of the prefetch prediction model 118, model updater 120, prefetch prediction engine 122, battery meter 124, and fetch queues 126 is explained in more detail with reference to Fig. 4. The scheduler 132 is explained in more detail with reference to Fig. 5.

Referring to Fig. 2, a message 150 sent by a server to a client in response to a client request typically includes a header 151 and a body 154. The header 151 stores information about the message, such as its source, content-type and encoding. In one preferred embodiment, the header 151 also includes an additional field 152 for storing prefetch hints. The prefetch hints are URLs or network addresses of files that the client computer may decide to prefetch. For instance, the client computer may prefetch one or more of the files represented by the prefetch hints 152 while decoding the file or other content in the body 154 of the message 150. The body 154 of the message contains the content of the message, such as a file or HTML page, and that content may include links or tags 156 that reference other files. The files referenced by the links or tags 156 are also candidates for prefetching by the client computer. In one embodiment the prefetch hints include a probability value for each URL or network address listed in the prefetch hints. The probability value indicates the predicted probability that the associated file will be requested by the client which sent the client request.

Prefetch Prediction Model

Referring to Fig. 3, the prefetch prediction model 118 in a preferred embodiment includes several types of parameters, such as energy usage parameters, user usage pattern parameters, network characteristics and so on. More specifically, the prefetch prediction model 118 preferably includes two or more of the following types of parameters:

- CPU energy usage parameters 160, for instance indicating CPU energy usage per prefetch, additional CPU energy used for downloading various types of files, CPU energy

during idle time, CPU fixed energy required to send a packet, CPU energy/byte required to send a packet, CPU fixed energy required to receive a packet, and CPU energy/byte required to receive a packet;

- Network interface energy usage parameters 162, for instance network interface fixed energy required to send a packet, network interface energy/byte (energy usage per byte) required to send a packet, network interface fixed energy required to receive a packet, network interface energy/byte (energy usage per byte) required to receive a packet, network interface energy to enter sleep mode, network interface time to enter sleep mode, network interface energy to exit sleep mode, network interface time to exit sleep mode, network interface power during idle mode, and network interface power during sleep mode;
- Memory access energy usage parameters 163, for instance memory read/write energy costs per byte or word, and energy costs of DMA operations;
- Energy supply parameters 164, for instance total energy availability, efficiency loss as a function of power drain, and energy efficiency as a function of power usage mode (e.g., as a function of level of power load, and pulsed mode vs continuous mode);
- User behavior and preference parameters 166, for instance estimate of user's apparent think time (estimate of mean or median time between a client's request for a file and the client's subsequent request for another file), preference for speed versus battery lifetime, preference for speed versus battery lifetime as a function of remaining battery lifetime, history of user's recent references, and prefetch-benefit threshold;
- Network characteristics parameters 168, for instance link bandwidth, current link utilization, and round-trip time to server;
- Client cache status 170, for instance the degree of utilization of a cache, allowable cache lifetime, cache entries, and stale cache entries; and
- Prefetch benefit determination parameters 172, for instance the estimated energy costs and benefits of doing a given prefetch or type of prefetch.

While the above description of the prefetch prediction model 118 refers to parameters concerning "a user," when the client computer system is a multi-user system the prefetch prediction model and mechanisms of the present preferably take into account the preferences and actions of multiple users of a client computer system.

Operation of Prefetch System Components

Referring to Fig. 4, prefetch model updater 120 receives usage statistics, such as prefetch utilization statistics (concerning the quantity and types of prefetched files which were used by the client computer prior to the prefetched files being evicted from the cache 134) and energy usage statistics (concerning the energy used by prefetch operations and the net energy saved or lost by prefetch operations), and uses that information to adjust prefetch prediction model 118 to changing conditions. Prefetch prediction engine 122 uses prefetch prediction model 118, along with user information and system status data to predict the net impact on energy usage by the client computer that would result from prefetching specified files in a list of prefetch candidates, and to determine which prefetch candidates to add to the prefetch queue 130. The information and data used by the prefetch prediction engine 122 for this purpose preferably includes at least two of the following types of information and data:

- user behavior and preferences,
- the content currently being viewed or used by the client,
- the rate at which the user has been requesting new files or pages,
- the status (e.g., fullness) of the client computer cache 134,
- the current energy level (battery meter reading) of the client computer;
- current network congestion;
- bandwidth availability between the client computer system and the server; and
- round-trip time from client to server.

In another embodiment, the prefetch prediction engine 122 predicts the net impact on energy usage by the overall system (including client, server and other system components) that would result from prefetching specified files from a list of prefetch candidates.

If an entry or item in the list of prefetch candidates identifies a specified file that is already in the demand fetch queue 128, the prefetch prediction engine preferably deletes that entry from the list of entries to be added to the prefetch queue 130. If an entry corresponding to a specified file is in cache 134, the prefetch prediction engine 122 does not add it to the prefetch queue 130, unless the entry in the cache is stale. If an entry corresponding to a

specified file is already in prefetch queue 130, the corresponding entry in prefetch queue may be removed or modified as appropriate.

Referring to Fig. 5, if there is a prefetch candidate available (step 200) and if the prefetch candidate is not in the cache, demand fetch queue, or prefetch queue (step 202), then a decision – based upon the candidate, status information and other data – is made as to whether to prefetch the candidate (step 204). As discussed above, the prefetch decision at step 204 is made by the prefetch prediction engine 122 using the prefetch prediction model 118 to determine if the benefits of prefetching the candidate are predicted to outweigh the costs, including energy usage associated with the prefetch operation. If the decision is to prefetch, the candidate is added to the prefetch queue (step 206). Optionally, the prefetch prediction engine may include queue pruning instructions for re-evaluating entries in the queue and flushing from the queue any entries in the queue deselected by the re-evaluating (step 208). For instance, the prefetch prediction engine may replace (at step 208) a least desirable entry in the prefetch queue with an entry for the candidate currently being processed.

In a preferred implementation, the prefetch queue is completely or partially flushed (at step 208) when the prefetch prediction engine determines that all or some of the queued prefetches are not likely to be useful. For example, such a flushing of the prefetch queue may be performed when an application in the client computer issues a request for an object that is not in the local cache, is not in the process of being demand fetched, and is not in the prefetch queue. If these conditions are satisfied (i.e., a miss occurs), then the entity (e.g., the prefetch prediction engine 122) that re-evaluates the prefetch queue (e.g., at step 208, Fig. 5) may decide that some or all of the pending prefetch entries correspond to a "path" that the client is unlikely to follow, and that therefore these entries should be deleted. On the other hand, despite the miss, the prefetch prediction engine may nevertheless continue to predict that some of the items in the prefetch queue will be required in the future, in which case those items are not flushed from the prefetch queue. The items remaining in the prefetch queue are prefetched when the client computer system is able to schedule those items to be fetched.

After the current candidate has been processed and a decision has been made with regard to that candidate, the prefetch candidate filtering process resumes at step 200.

The prefetch queue 130 may be reevaluated and modified (step 208) at any time prior to scheduling downloading of the files listed in the prefetch queue (step 214).

A scheduler 132 (Fig. 1) reads the entries in the prefetch queue 130 and demand fetch queue 128 and schedules the downloading of the files listed in those entries (step 214). Typically, entries in the demand fetch queue 128 are given higher priority by the scheduler than entries in the prefetch queue 130. The files scheduled for downloading are fetched (step 216) in accordance with their scheduling for use by the client computer and/or for storage in the cache 134 for possible later use by the client computer.

Server Computer Prefetch Prediction

Referring to Fig. 6, a server computer system (server) 220 is shown in accordance with the present invention. The server computer system 220 includes memory 230 for storing applications executable by one or more central processing units 222 and data structures such files and the like. A network interface 224 allows communication through communications network(s) 140 with clients (client computer systems) 100.

In one embodiment, memory 230, which typically includes high speed random access memory as well as non-volatile storage such as disk storage, stores:

- an operating system 232, for providing basic system services;
- a file system 234, which may be part of the operating system;
- application programs 236;
- a server module 238, for receiving and processing client requests, such as a request for a specified file;
- a prefetch predictor 240, for identifying files for possible prefetching by a client 228;
- a prefetch efficiency model 242;
- a prefetch hint pruner 244, for utilizing the prefetch efficiency model to prune the set of identified files so as to improve energy efficiency in prefetching and to eliminate

prefetches that the system decides, upon re-evaluation, are not likely to be requested by the client.

The interaction of the server module 238, prefetch predictor 240, prefetch efficiency model 242, and prefetch hint pruner 244 is explained in more detail with reference to Fig. 8.

Referring to Fig. 7, the server-side prefetch efficiency model 242 in a preferred embodiment includes several types of parameters, such as transmission thresholds, accuracy statistics and so on. Some of the parameters in the server-side prefetch efficiency model 242 have values that change dynamically in accordance with changes in the operating environment of the server computer system. The prefetch efficiency model 242 preferably includes one or more of the following types of parameters for use by the server-side prefetch predictor 240:

- transmission thresholds 250, for instance the maximum number of hints to send per reply, and the maximum size (in bytes) of hints to send per reply; and
- prefetch accuracy statistics 254 (e.g., statistics received from one or more client computers).

In addition, the prefetch efficiency model 242 preferably includes one or more of the following types of parameters for use by the server-side prefetch pruner 244:

- client attribute parameters 256, for instance parameters sent by the client from the client's prefetch prediction model;
- network attribute parameters 258, for instance the bandwidth and congestion level of the network connection between the server and each client with which it has a current client-server session; the network attribute parameters 258 enable the server's prefetch pruner to determine an estimated data transfer time for each prefetch candidate; and
- a hint log 260, for instance a log of the URLs or network addresses for which the server has sent prefetch hints to each client during a current client-server session; the log may also include a record of the files fetched by each client, and may further indicate the time that the files were fetched so as to enable the server to determine which of those files may be considered to be stale (i.e., beyond the allowed cache lifetime).

Referring to Fig. 8, server module 238 receives a request from client computer for a specified file. Prefetch predictor 240, utilizing data provided by the server module 238, identifies additional files for possible prefetching. Prefetch hint pruner 244, before forwarding the additional files to the server module 238, prunes the list of identified additional files in accordance with prefetch efficiency model 242. The pruning consists of removing from the list references to one or more of the files that do not meet the requirements of the prefetch efficiency model 242. The server module 238 then sends a reply to the client computer. The reply includes a content portion (see Fig. 2) containing the specified file and a supplemental portion containing prefetch hint information identifying at least one of the additional files.

Referring to Fig. 9, a request message 300 sent by a client to a server typically includes a header 301 and a body 304. The header 301 stores information about the message, such as its destination and encoding. In one preferred embodiment, the header 301 also includes an additional field 302 for storing client hints. The client hints are a subset of the parameters of the client's prefetch prediction model, or parameters derived at least in part from the client's prefetch prediction model. For instance, the client hints 302 may include the client's costs for sending and receiving data bytes and packets and the client's threshold value for predicted prefetch probabilities. The body 304 of the request message contains the client request, such as the URL or network address of the document being requested by the client and parameters (if any) for specifying dynamic content to be produced by the server while responding to the client request.

Alternate Embodiments

In an alternate embodiment, the prefetch prediction model (client computer system, Fig. 1) or prefetch efficiency model (server computer system, Fig. 6) includes cost prediction parameters not based on energy usage or not solely based on energy usage. In this alternate embodiment, it is advantageous to reduce the monetary charges incurred by the client's network activity. For instance, the prefetch prediction model may include one or more parameters that take into account network usage charges associated with each packet or message transmission, or based on the number of bytes transmitted, without regard to the amount of energy used. In some implementations, the use of predictive prefetching may

increase the monetary charges incurred by the client computer system. In these implementations the role of the prefetch prediction engine and prefetch prediction model are to ensure that only the most cost efficient prefetches are performed. One or more threshold parameters (e.g., a threshold probability that each prefetched file will be used by the client) in the prefetch prediction model are based on the amount of additional cost the client is willing to incur, on average, in exchange for the latency reduction obtained by prefetching files predicted to be needed by the client in the future.

In another alternate embodiment, a client computer system has multiple wireless network interfaces (e.g., radios) that operate simultaneously, each having different energy/byte and latency/byte costs. In such an embodiment, the client computer system may be configured to use the wireless interface with the lowest energy/byte for prefetches of files that are not likely to be required by the client immediately or that are less likely to be used by the client, and to otherwise use a wireless interface with higher energy/byte costs (and presumably lower latency/byte).

The present invention can be implemented as a computer program product that includes a computer program mechanism embedded in a computer readable storage medium. For instance, the computer program product could contain the program modules shown in Figs. 1 or 6 or both. These program modules may be stored on a CD-ROM, magnetic disk storage product, or any other computer readable data or program storage product. The software modules in the computer program product may also be distributed electronically, via the Internet or otherwise, by transmission of a computer data signal (in which the software modules are embedded) on a carrier wave.

While the present invention has been described with reference to a few specific embodiments, the description is illustrative of the invention and is not to be construed as limiting the invention. Various modifications may occur to those skilled in the art without departing from the true spirit and scope of the invention as defined by the appended claims.